Discrete Applied Mathematics 158 (2010) 627–635

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Modified algorithms for the minimum volume enclosing axis-aligned ellipsoid problem

Wei-jie Cong *, Hong-wei Liu

Department of Applied Mathematics, Xidian University, Xi'an 710071, PR China

ARTICLE INFO

Article history:
Received 22 October 2009
Received in revised form 7 December 2009
Accepted 8 December 2009
Available online 23 December 2009

Keywords:
Minimum volume ellipsoids
Axis-aligned ellipsoids
Core sets
Approximation algorithms
Complexity analysis

ABSTRACT

Consider the problem of computing a $(1+\epsilon)$-approximation to the minimum volume axis-aligned ellipsoid (MVAE) enclosing a set of m points in $\mathbb{R}^n$. We first provide an extension and improvement to algorithm proposed in Kumar and Yıldırım (2008) [5] (the KY algorithm) for the MVAE problem. The main challenge of the MVAE problem is that there is no closed form solution in the line search step (beta). Therefore, the KY algorithm proposed a certain choice of beta that leads to their complexity and core set results in solving the MVAE problem. We further analyze the line search step to derive a new beta, relying on an analysis of up to the fourth order derivative. This choice of beta leads to the improved complexity and core set results. The second modification is given by incorporating "away steps" into the first one at each iteration, which obtains the same complexity and core set results as the first one. In addition, since the second modification uses the idea of "dropping points", it has the potential to compute smaller core sets in practice. Some numerical results are given to show the efficiency of the modified algorithms.

1. Introduction

Given a set of points $S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$, in this paper, we consider the problem of computing an approximate minimum volume axis-aligned ellipsoid (MVAE) enclosing S, denoted by MVAE(S).

A full-dimensional axis-aligned ellipsoid $E_{D,c} \subset \mathbb{R}^n$ is specified by a positive definite diagonal matrix $D = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n\times n}$ and a center $c = (c_1, \ldots, c_n)^T \in \mathbb{R}^n$ and is defined as [5]

$$E_{D,c} := \{x \in \mathbb{R}^n : (x-c)^T D(x-c) \le 1\} = \left\{x \in \mathbb{R}^n : \sum_{j=1}^{n} \left[\sqrt{d_j}(x_j - c_j)\right]^2 \le 1\right\}. \tag{1}$$

The scaled volume of $E_{D,c}$ is given by

$$\mathrm{Vol}(E_{D,c}) = \det D^{-1/2} = \prod_{j=1}^{n} \left(1/\sqrt{d_j}\right). \tag{2}$$

Given $\epsilon > 0$, $E_{D,c}$ is said to be a $(1+\epsilon)$-approximation to MVAE(S) if

$$S \subseteq E_{D,c}, \qquad \mathrm{Vol}(E_{D,c}) \le (1+\epsilon)\mathrm{Vol}(\mathrm{MVAE}(S)). \tag{3}$$

* Corresponding author. Fax: +86 029 88206278.
E-mail address: cong24518@163.com (W.-j. Cong).

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.12.003

In addition, a subset $X \subseteq S$ is called an ϵ-core set (or a core set) of S if there exists an axis-aligned ellipsoid $E_{D,c} \subset \mathbb{R}^n$ such that $S \subseteq E_{D,c}$ and $E_{D,c}$ is a $(1+\epsilon)$-approximation to MVAE(X). The reader is referred to [5] for a more detailed account of background knowledge of the MVAE problem.

From [5], the problem of computing MVAE(S) can be formulated as the following convex optimization problem:

$$\min_{\gamma,\mu} -\sum_{j=1}^{n} \log \gamma_j : \sum_{j=1}^{n} (\gamma_j x_j^i - \mu_j)^2 \le 1, \quad i = 1, \ldots, m, \tag{4}$$

where $\gamma = (\gamma_1, \ldots, \gamma_n)^T \in \mathbb{R}^n$ and $\mu = (\mu_1, \ldots, \mu_n)^T \in \mathbb{R}^n$ are the primal variables. The Lagrangian dual of (4) is given by

$$\max_{\sigma} \frac{n}{2} \log n + \frac{1}{2} \sum_{j=1}^{n} \log(u_j(\sigma) - v_j^2(\sigma)) : \mathrm{e}^T \sigma = 1, \quad \sigma \ge 0, \tag{5}$$

where $\sigma = (\sigma_1, \ldots, \sigma_m)^T \in \mathbb{R}^m$ is the dual variable, $e \in \mathbb{R}^m$ denotes the vector of all ones, and

$$u_j(\sigma) := \sum_{i=1}^{m} \sigma_i (x_j^i)^2, \qquad v_j(\sigma) := \sum_{i=1}^{m} \sigma_i x_j^i, \quad j = 1, \ldots, n. \tag{6}$$

It follows from [5, Lemma 2.1], that if (γ^*, μ^*) and σ^* denote the optimal solutions of (4) and (5), respectively, then MVAE(S) can be given by

$$\mathrm{MVAE}(S) = \{x \in \mathbb{R}^n : (x - c^*)^T D^* (x - c^*) \le 1\} = \left\{ x \in \mathbb{R}^n : \sum_{j=1}^{n} \left[\sqrt{d_j^*} (x_j - c_j^*) \right]^2 \le 1 \right\}, \tag{7}$$

where

$$c_j^* := \frac{\mu_j^*}{\gamma_j^*} = v_j(\sigma^*), \qquad d_j^* := (\gamma_j^*)^2 = \frac{1}{n(u_j(\sigma^*) - v_j^2(\sigma^*))}, \quad j = 1, \ldots, n. \tag{8}$$

Therefore, MVAE(S) can be computed by solving the dual problem (5), which will be the basis of the algorithm proposed in Kumar and Yıldırım [5] (the KY algorithm) and our modifications.

In [5], the KY algorithm computes an approximate solution to the dual problem (5). This algorithm is essentially a first order algorithm rooted from the Khachiyan algorithm [3] for the minimum volume enclosing ellipsoid (MVEE) problem [4], which, by itself, is the adaptation of Frank–Wolfe algorithm [2]. In contrast with the MVEE problem, the main difference and challenge in applying the Khachiyan algorithm to the MVAE problem is that there is no closed form solution in the line search step (beta) (see (12) in [5]). Therefore, the KY algorithm proposed a certain choice of beta, which computes a $(1+\epsilon)$-approximation to MVAE(S) in

$$O(mn^2(\log n + n^2[(1+\epsilon)^{2/n} - 1]^{-1})) \tag{9}$$

arithmetic operations, which reduces to $O(mn^5/\epsilon)$ for $\epsilon \in (0, 1)$. Their analysis establishes that there exists a core set of size $O(n^4/\epsilon)$ for $\epsilon \in (0, 1)$. In addition, their paper mentioned that compared with the MVEE problem, such theoretical results of the complexity and the core set are rather pessimistic, which cannot be improved in general. One of the potential reasons is the utilization of the lower bound instead of the exact maximizer of the line search problem.

In this paper, we first provide an extension and improvement to the KY algorithm. We further analyze the line search step to derive a new beta, relying on an analysis of up to the fourth order derivative (see, Lemma 2.1). This choice of beta leads to the improved complexity result as follows:

$$O(mn^2(\log n + [(1+\epsilon)^{2/n} - 1]^{-1})), \tag{10}$$

which reduces to $O(mn^3/\epsilon)$ for $\epsilon \in (0, 1)$. Our analysis returns a core set of size $O(n^2/\epsilon)$ for $\epsilon \in (0, 1)$. They are consistent with the complexity and core set results of the MVEE problem [4]. Then, our second modification is given by incorporating "away steps" into the first one, which obtains the same complexity and core set results as the first one. In addition, since the second modification uses the idea of "dropping points" [7], it has the potential to compute smaller core sets in practice. Finally, the numerical results reveal the efficiency of the modified algorithms.

2. The first modification and analysis

In this section, we give our first modification that computes a $(1+\epsilon)$-approximation to MVAE(S). Then, we present a detailed analysis of this modified algorithm.

2.1. The first modified algorithm

We describe the first modified algorithm as follows:

Algorithm 2.1. Input the set of points $S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$, $\epsilon > 0$. Set $k = 0$.

Step 1: Run Initial Volume Approximation Algorithm [5] on S to obtain X_0, set $X = X_0$.
Step 2: Let $\sigma^0 \in \mathbb{R}^m$ be such that $\sigma_i^0 = 1/|X_0|$ for $x^i \in X_0$ and $\sigma_i^0 = 0$ otherwise.
Step 3: Set $E_k := \left\{x \in \mathbb{R}^n : \sum_{j=1}^n \frac{(x_j - v_j(\sigma^k))^2}{n(u_j(\sigma^k) - v_j^2(\sigma^k))} \le 1\right\}$ and $w_j^i(\sigma^k) := \frac{(x_j^i - v_j(\sigma^k))^2}{n(u_j(\sigma^k) - v_j^2(\sigma^k))}$.
Step 4: Compute $i_+ := \arg\max\{\sum_{j=1}^n w_j^i(\sigma^k) : i = 1, \ldots, m\}$ and $\epsilon_k := \sum_{j=1}^n w_j^{i_+}(\sigma^k) - 1$.
If $\epsilon_k \le (1+\epsilon)^{2/n} - 1$, stop, output σ^k, X and $\sqrt{1+\epsilon_k}E_k$. Otherwise, go to Step 5.
Step 5: Set $X := X \cup \{x^{i_+}\}$; $\hat{\beta}_k := \frac{\epsilon_k}{1+n\sum_{j=1}^n (w_j^{i_+}(\sigma^k))^2}$, $\sigma^{k+1} := (1-\hat{\beta}_k)\sigma^k + \hat{\beta}_k e^{i_+}$. Let $k := k+1$ and go to Step 3.

2.2. Analysis of the first modified algorithm

In this subsection, we first simply describe Algorithm 2.1. Given a point set S, Algorithm 2.1 uses the same initialization procedure as the KY algorithm [5] to obtain an initial core set X_0 and an initial feasible solution σ^0. It follows from the KY algorithm that when Algorithm 2.1 is terminated, it can obtain a $(1+\epsilon)$-approximation to MVAE(S). The main difference between the two algorithms is the choice of the lower bound of β_k^* (see Lemma 2.1). The KY algorithm computes a $(1+\epsilon)$-approximation to MVAE(S) by using $\beta_k = \epsilon_k/[(n+1)(1+\epsilon_k)]$, whose complexity bound is given by (9). Algorithm 2.1 uses $\hat{\beta}_k$ given by Step 5, which yields an improved complexity bound (10).

In the following, a more detailed analysis of Algorithm 2.1 will be presented. We first describe how $\hat{\beta}_k$ is obtained at the kth iteration of Algorithm 2.1. Then, we establish the improved complexity and core set results.

Let $g(\sigma) := \frac{n}{2}\log n + \frac{1}{2}\sum_{j=1}^n \log(u_j(\sigma) - v_j^2(\sigma))$. It follows from (17) in [5] that

$$g((1-\beta)\sigma^k + \beta e^{i_+}) = g(\sigma^k) + \Delta_k(\beta),$$

where

$$\Delta_k(\beta) := \frac{n}{2}\log(1-\beta) + \frac{1}{2}\sum_{j=1}^n \log(1 + \beta n w_j^{i_+}(\sigma^k)), \quad k = 0, 1, \ldots. \tag{11}$$

From [5, Lemma 4.3], we obtain the fact that the function $\Delta_k(\beta)$ has a unique maximizer $\beta_k^* \in [0, 1)$, which does not have a closed form. Thus, the KY algorithm presented a lower bound $\beta_k = \epsilon_k/[(n+1)(1+\epsilon_k)]$ instead of the exact maximizer β_k^*. The following lemma provides another lower bound $\hat{\beta}_k$ of the maximizer β_k^* at each iteration.

Lemma 2.1. *Let β_k^* be the unique maximizer of $\Delta_k(\beta)$ in $[0, 1)$. Then, we have*

$$\beta_k^* \ge \hat{\beta}_k := \frac{\epsilon_k}{1 + n\sum_{j=1}^n (w_j^{i_+}(\sigma^k))^2}, \quad k = 0, 1, \ldots, \tag{12}$$

where ϵ_k and $w_j^{i_+}(\sigma^k)$ are defined as in Algorithm 2.1.

Proof. By (11), we obtain that

$$\Delta_k'(\beta) = -\frac{n}{2(1-\beta)} + \frac{1}{2}\sum_{j=1}^n \frac{n w_j^{i_+}(\sigma^k)}{1 + \beta n w_j^{i_+}(\sigma^k)}, \tag{13a}$$

$$\Delta_k''(\beta) = -\frac{n}{2(1-\beta)^2} - \frac{1}{2}\sum_{j=1}^n \frac{n^2 (w_j^{i_+}(\sigma^k))^2}{(1 + \beta n w_j^{i_+}(\sigma^k))^2}, \tag{13b}$$

$$\Delta_k'''(\beta) = -\frac{n}{(1-\beta)^3} + \sum_{j=1}^n \frac{n^3 (w_j^{i_+}(\sigma^k))^3}{(1 + \beta n w_j^{i_+}(\sigma^k))^3}, \tag{13c}$$

$$\Delta_k^{(4)}(\beta) = -\frac{3n}{(1-\beta)^4} - 3\sum_{j=1}^n \frac{n^4 (w_j^{i_+}(\sigma^k))^4}{(1 + \beta n w_j^{i_+}(\sigma^k))^4}. \tag{13d}$$

It follows from [5, Lemma 4.3] that there exists a unique $\beta_k^* \in [0, 1)$ such that $\Delta_k'(\beta_k^*) = 0$. Similarly, since $\Delta_k^{(4)}(\beta)$ is negative on $[0, 1)$, $\Delta_k'''(\beta)$ is strictly decreasing on $[0, 1)$. By (13c), we obtain

$$\lim_{\beta\uparrow 1} \Delta_k'''(\beta) = -\infty \quad \text{and} \quad \Delta_k'''(0) = -n + n^3 \sum_{j=1}^{n} (w_j^{i_+}(\sigma^k))^3 \geq -n + n\left(\sum_{j=1}^{n} w_j^{i_+}(\sigma^k)\right)^3 = -n + n(1+\epsilon_k)^3 \geq 0,$$

where we used the following inequality for $\lambda = 3$ to derive the first inequality

$$\left(\frac{a_1^\lambda + a_2^\lambda + \cdots + a_n^\lambda}{n}\right)^{1/\lambda} \geq \frac{a_1 + a_2 + \cdots + a_n}{n}, \quad \lambda = 1, 2, \ldots \quad \text{and} \quad a_i \geq 0, \quad i = 1, \ldots, n. \tag{14}$$

Therefore, it follows that there exists a unique $\hat{\beta}_k^* \in [0, 1)$ such that $\Delta_k'''(\hat{\beta}_k^*) = 0$. In addition, note that $\hat{\beta}_k^*$ is the inflexion of the function $\Delta_k'(\beta)$.

From (13c), we have

$$\begin{aligned}
\Delta_k'''(\beta_k^*) &= -\frac{n}{(1-\beta_k^*)^3} + \sum_{j=1}^{n} \left(\frac{n w_j^{i_+}(\sigma^k)}{1 + \beta_k^* n w_j^{i_+}(\sigma^k)}\right)^3 \\
&\geq -\frac{n}{(1-\beta_k^*)^3} + \frac{1}{n^2}\left(\sum_{j=1}^{n} \frac{n w_j^{i_+}(\sigma^k)}{1 + \beta_k^* n w_j^{i_+}(\sigma^k)}\right)^3 \\
&= -\frac{n}{(1-\beta_k^*)^3} + \frac{1}{n^2}\left(\frac{n}{1-\beta_k^*}\right)^3 \\
&= 0 = \Delta_k'''(\hat{\beta}_k^*),
\end{aligned}$$

where we used (14) and $\Delta_k'(\beta_k^*) = 0$ to derive the inequality and the second equality, respectively. Since $\Delta_k'''(\beta)$ is strictly decreasing on $[0, 1)$, we obtain $\beta_k^* \leq \hat{\beta}_k^*$. Therefore, $\Delta_k'(\beta)$ is a strictly convex function on $[0, \beta_k^*]$.

The tangent equation of $\Delta_k'(\beta)$ at point $(0, \Delta_k'(0))$ is given by

$$f(x) = \Delta_k''(0)x + \Delta_k'(0). \tag{15}$$

When $f(x) = 0$, we obtain $x = -\Delta_k'(0)/\Delta_k''(0) \leq \beta_k^*$, which just is the selection of $\hat{\beta}_k$ in Algorithm 2.1. Thus, we have

$$\beta_k^* \geq \hat{\beta}_k = -\frac{\Delta_k'(0)}{\Delta_k''(0)} = -\frac{n\epsilon_k/2}{-n/2 - (n^2/2)\sum_{j=1}^{n}(w_j^{i_+}(\sigma^k))^2} = \frac{\epsilon_k}{1 + n\sum_{j=1}^{n}(w_j^{i_+}(\sigma^k))^2}, \quad k = 0, 1, \ldots.$$

This completes the proof of (12). □

The following lemma gives a lower bound on the improvement of $\Delta_k(\beta)$ by using $\hat{\beta}_k$ given in Lemma 2.1.

Lemma 2.2. *Let $n \geq 2$, β_k^* be the unique maximizer of $\Delta_k(\beta)$ in $[0, 1)$ and $\hat{\beta}_k$ be given by (12). Then, for $k = 0, 1, \ldots$, we have*

$$\Delta_k(\beta_k^*) \geq \Delta_k(\hat{\beta}_k) \geq \begin{cases} 1/18, & \text{if } \epsilon_k \geq 1, \\ \epsilon_k^2/18, & \text{if } \epsilon_k < 1. \end{cases} \tag{16}$$

Proof. Since β_k^* is the unique maximizer of $\Delta_k(\beta)$ in $[0, 1)$, we have

$$\Delta_k(\beta_k^*) \geq \Delta_k(\hat{\beta}_k) = \Delta_k(\hat{\beta}_k) - \Delta_k(0) = \int_0^{\hat{\beta}_k} \Delta_k'(x)\mathrm{d}x \geq \int_0^{\hat{\beta}_k} (\Delta_k''(0)x + \Delta_k'(0))\mathrm{d}x = \frac{1}{2}\hat{\beta}_k \Delta_k'(0),$$

where we used Newton–Leibnitz formula, (15) and geometric significance of definite integral to derive the second equality, the second inequality and the last equality, respectively.

By (12) and $\Delta_k'(0) = -n/2 + n(1+\epsilon_k)/2 = n\epsilon_k/2$, we obtain

$$\Delta_k(\hat{\beta}_k) \geq \frac{1}{2}\frac{\epsilon_k}{1 + n\sum_{j=1}^{n}(w_j^{i_+}(\sigma^k))^2}\frac{n\epsilon_k}{2} = \frac{1}{4}\frac{\epsilon_k^2}{(1/n) + \sum_{j=1}^{n}(w_j^{i_+}(\sigma^k))^2} \geq \frac{1}{4}\frac{\epsilon_k^2}{0.5 + (1+\epsilon_k)^2},$$

where we used $n \geq 2$ and $\sum_{j=1}^{n}(w_j^{i_+}(\sigma^k))^2 \leq (\sum_{j=1}^{n} w_j^{i_+}(\sigma^k))^2 = (1+\epsilon_k)^2$ to derive the last inequality.

It is easy to verify that the function $f_1(x) := x^2/[0.5+(1+x)^2]$ is strictly increasing for $x \geq 0$, and $f_2(x) := 1/[0.5+(1+x)^2]$ is strictly decreasing for $x \geq 0$. Thus, we have $f_1(x) \geq 2/9$ for $x \geq 1$ and $f_2(x) \geq 2/9$ for $0 \leq x \leq 1$. Therefore, we obtain

$$\Delta_k(\hat{\beta}_k) \geq \begin{cases} 1/18, & \text{if } \epsilon_k \geq 1, \\ \epsilon_k^2/18, & \text{if } \epsilon_k < 1. \end{cases}$$

This completes the proof of (16). □

Remark 2.1. It is interesting that the KY step size [5] can be longer than that proposed here when ϵ_k is large, e.g., if one $w_j^{i+}(\sigma^k)$ is $1+\epsilon_k$ and the others all zero and ϵ_k is bigger than $1/n$. Indeed, the guaranteed reduction when ϵ_k is large, $1/18$, is smaller than that given by KY when delta is large, $(1/2)\log 2 - 1/4$. But this is not too important, since they do not affect this part $O(n \log n)$ of the iteration complexity results (see, Theorem 2.1). However, if ϵ_k is small and all individual $w_j^{i+}(\sigma^k) = (1+\epsilon_k)/n, j = 1, \ldots, n$, KY's beta is about $\epsilon_k/(n+1)$ and our paper's around ϵ_k. It follows from Lemmas 4.4 and 4.5 of [5] that $\Delta_k(\beta_k) \geq \delta_k^2/16$ and $\delta_k = \epsilon_k/(n+1)$. In this case, Lemma 4.4 can obtain $\Delta_k(\beta_k) \geq \epsilon_k^2/16(n+1)^2$. Therefore, it is easy to verify that these results lead to differences in the final complexity results.

The following theorem establishes the iteration complexity of Algorithm 2.1.

Theorem 2.1. *Let $n \geq 2, \epsilon > 0$. Algorithm 2.1 computes a $(1+\epsilon)$-approximation to* MVAE(S) *in $O(n(\log n + [(1+\epsilon)^{2/n} - 1]^{-1}))$ iterations.*

Proof. From Step 3 of Algorithm 2.1, a trial axis-aligned ellipsoid E_k is constructed at the kth iteration and is given by

$$E_k = \left\{ x \in \mathbb{R}^n : \sum_{j=1}^{n} \frac{(x_j - v_j(\sigma^k))^2}{n(u_j(\sigma^k) - v_j^2(\sigma^k))} \leq 1 \right\}, \quad k = 0, 1, \ldots.$$

Therefore, we obtain from (2) that

$$\log \mathrm{Vol}(E_k) = \log \prod_{j=1}^{n} \sqrt{n(u_j(\sigma^k) - v_j^2(\sigma^k))} = \frac{n}{2} \log n + \frac{1}{2} \sum_{j=1}^{n} \log(u_j(\sigma^k) - v_j^2(\sigma^k)) = g(\sigma^k).$$

Obviously, if σ^k is an optimal solution σ^* of (5), then E_k is the optimal axis-aligned ellipsoid $E^* :=$ MVAE(S) (see, (7) and (8)). It follows from [5, Lemma 4.2] that

$$\log \mathrm{Vol}(E^*) \leq \log \mathrm{Vol}(E_0) + n \log n + (n/2) \log 2 \leq \log \mathrm{Vol}(E_0) + (3n/2) \log n \quad \text{for } n \geq 2.$$

Thus, we have $\log \mathrm{Vol}(E^*) - \log \mathrm{Vol}(E_0) = g(\sigma^*) - g(\sigma^0) \leq (3n/2) \log n$.

Let us first consider the iteration with $\epsilon_k \geq 1$. It follows from Lemma 2.2 that $\log \mathrm{Vol}(E_{k+1}) - \log \mathrm{Vol}(E_k) = g(\sigma^{k+1}) - g(\sigma^k) = \Delta_k(\hat{\beta}_k) \geq 1/18$. Therefore, Algorithm 2.1 needs at most $K \leq (3n/2) \log n/(1/18) = O(n \log n)$ iterations to compute a solution σ^k with $\epsilon_k < 1$.

Let us now consider the iteration with $\epsilon_k < 1$. From [3, Lemma 4], Algorithm 2.1 needs at most $O(n/\eta)$ iterations to obtain $\epsilon_k \leq \eta$ for any $\eta \in (0, 1)$. Note that $\eta = (1+\epsilon)^{2/n} - 1$ in Algorithm 2.1. In addition, it follows from [5, Theorem 4.1] that when the termination criterion is satisfied, Algorithm 2.1 returns a $(1+\epsilon)$-approximation to MVAE(S). This completes the proof. □

It is easy to show that each iteration of Algorithm 2.1 requires $O(mn)$ arithmetic operations. Therefore, combining Theorem 2.1 and [5, Theorem 4.3], the following corollary establishes the improved complexity and core set results to compute a $(1+\epsilon)$-approximation to MVAE(S).

Corollary 2.2. *Let $n \geq 2, \epsilon > 0$. Algorithm 2.1 computes a $(1+\epsilon)$-approximation to* MVAE(S) *in $O(mn^2(\log n + [(1+\epsilon)^{2/n} - 1]^{-1}))$ arithmetic operations and returns an ϵ-core set $X \subseteq S$ such that $|X| = O(n(\log n + [(1+\epsilon)^{2/n} - 1]^{-1}))$.*

Remark 2.2. Since $[(1+\epsilon)^{2/n} - 1]^{-1} = O(n/\epsilon)$ for $\epsilon \in (0, 1)$, the complexity bound of Algorithm 2.1 for computing a $(1+\epsilon)$-approximation to MVAE(S) can reduce to $O(mn^3/\epsilon)$ for $\epsilon \in (0, 1)$. In addition, the core set size reduces to $O(n^2/\epsilon)$ for $\epsilon \in (0, 1)$. They are consistent with the complexity and core set results of the MVEE problem [4]. In addition, it is easy to see that the simple Frank–Wolfe method [2] is really justified by the core set results. Without the axis-aligned restriction, Frank–Wolfe steps are necessary to keep the cost of each iteration low because of rank-one updating formulae. In the axis-aligned case, the only reason for using such a crude method is to keep low cardinality core sets.

3. The second modification and analysis

In this section, we present and analyze our second modification that computes a $(1+\epsilon)$-approximation to MVAE(S). This modification is obtained by incorporating "away steps" into the first modification. A similar algorithm has recently been proposed for the MVEE problem [7].

3.1. The second modified algorithm

We describe the second modified algorithm as follows:

Algorithm 3.1. Input the set of points $S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n, \epsilon > 0$. Set $k = 0$.

Step 1: Run Initial Volume Approximation Algorithm [5] on S to obtain X_0, set $X = X_0$.
Step 2: Let $\sigma^0 \in \mathbb{R}^m$ be such that $\sigma_i^0 = 1/|X_0|$ for $x^i \in X_0$ and $\sigma_i^0 = 0$ otherwise.
Step 3: Set $E_k := \left\{x \in \mathbb{R}^n : \sum_{j=1}^n \frac{(x_j - v_j(\sigma^k))^2}{n(u_j(\sigma^k) - v_j^2(\sigma^k))} \le 1\right\}$ and $w_j^i(\sigma^k) := \frac{(x_j^i - v_j(\sigma^k))^2}{n(u_j(\sigma^k) - v_j^2(\sigma^k))}$.
Step 4: Compute $i_+ := \arg\max\{\sum_{j=1}^n w_j^i(\sigma^k) : i = 1, \ldots, m\}$ and $\epsilon_+ := \sum_{j=1}^n w_j^{i_+}(\sigma^k) - 1$;

$$i_- := \arg\min\left\{\sum_{j=1}^n w_j^i(\sigma^k) : i = 1, \ldots, m, \sigma_i^k > 0\right\} \quad \text{and} \quad \epsilon_- := 1 - \sum_{j=1}^n w_j^{i_-}(\sigma^k);$$

$$\epsilon_k := \max\{\epsilon_+, \epsilon_-\}.$$

If $\epsilon_k \le (1+\epsilon)^{2/n} - 1$, stop, output σ^k, X and $\sqrt{1+\epsilon_k}E_k$. Otherwise, go to Step 5.
Step 5: If $\epsilon_k = \epsilon_+$, go to Step 6. Otherwise, go to Step 7.
Step 6: Set $X := X \cup \{x^{i_+}\}$; $\hat\beta_k := \frac{\epsilon_k}{1+n\sum_{j=1}^n (w_j^{i_+}(\sigma^k))^2}$, $\sigma^{k+1} := (1-\hat\beta_k)\sigma^k + \hat\beta_k e^{i_+}$. Let $k := k+1$ and go to Step 3.
Step 7: Set $w_{j_+}^{i_-}(\sigma^k) := \max\{w_j^{i_-}(\sigma^k) : j = 1, \ldots, n\}$; $\hat\beta_k := \min\left\{\frac{\epsilon_k}{1-\epsilon_k+nw_{j_+}^{i_-}(\sigma^k)}, \frac{\sigma_{i_-}^k}{1-\sigma_{i_-}^k}\right\}$; If $\hat\beta_k = \frac{\sigma_{i_-}^k}{1-\sigma_{i_-}^k}$, then $X := X \setminus \{x^{i_-}\}$; $\sigma^{k+1} := (1+\hat\beta_k)\sigma^k - \hat\beta_k e^{i_-}$. Let $k := k+1$ and go to Step 3.

3.2. Analysis of the second modified algorithm

In this subsection, we first simply describe Algorithm 3.1. Algorithm 3.1 starts off with the same initialization procedure as Algorithm 2.1. In contrast with Algorithm 2.1, Algorithm 3.1 computes not only the farthest point $x^{i_+} \in S$ in terms of the ellipsoidal norm induced by E_k (see, Step 3) but also the closest point $x^{i_-} \in S$ among those with σ_i^k positive (see, Step 4). In Step 7 ("an away step"), we should have from [7] that

$$\beta_k^* := \arg \max_{\beta \in \left[0, \frac{\sigma_{i_-}^k}{1-\sigma_{i_-}^k}\right]} g((1+\beta)\sigma^k - \beta e^{i_-}). \tag{17}$$

For "an away step", $g((1+\beta)\sigma^k - \beta e^{i_-}) = g(\sigma^k) + \Delta_k(\beta)$, where

$$\Delta_k(\beta) := \frac{n}{2}\log(1+\beta) + \frac{1}{2}\sum_{j=1}^n \log(1 - \beta n w_j^{i_-}(\sigma^k)), \quad k = 0, 1, \ldots. \tag{18}$$

Compared with "an away step" for the MVEE problem [7], the main challenge here is still no closed form solution in the line search step (17).

The following lemma shows that the problem (18) has a unique maximizer β_k^* and provides a lower bound $\hat\beta_k$ of the maximizer β_k^*.

Lemma 3.1. *In "an away step", the problem (18) has a unique maximizer $\beta_k^* \in [0, 1/(nw_{j_+}^{i_-}(\sigma^k)))$. Furthermore, we have*

$$\beta_k^* \ge \hat\beta_k := \min\left\{\frac{\epsilon_k}{1-\epsilon_k+nw_{j_+}^{i_-}(\sigma^k)}, \frac{\sigma_{i_-}^k}{1-\sigma_{i_-}^k}\right\}, \quad k = 0, 1, \ldots, \tag{19}$$

where ϵ_k and $w_{j_+}^{i_-}(\sigma^k)$ are defined as in Algorithm 3.1.

Proof. By (18), we obtain that

$$\Delta_k'(\beta) = \frac{n}{2(1+\beta)} - \frac{1}{2}\sum_{j=1}^n \frac{nw_j^{i_-}(\sigma^k)}{1-\beta n w_j^{i_-}(\sigma^k)}, \tag{20a}$$

$$\Delta_k''(\beta) = -\frac{n}{2(1+\beta)^2} - \frac{1}{2}\sum_{j=1}^n \frac{n^2(w_j^{i_-}(\sigma^k))^2}{(1-\beta n w_j^{i_-}(\sigma^k))^2}. \tag{20b}$$

Clearly, $\Delta_k''(\beta)$ is negative for $\beta \geq 0$. Therefore, we obtain that $\Delta_k(\beta)$ is a strictly concave function and $\Delta_k'(\beta)$ is strictly decreasing for $\beta \geq 0$. In addition, we obtain from (20a) that $\Delta_k'(0) = \frac{n}{2} - \frac{n}{2}(1-\epsilon_k) = \frac{n\epsilon_k}{2} \geq 0$ and $\lim_{\beta \uparrow 1/(nw_{j_+}^{i_-}(\sigma^k))} \Delta_k'(\beta) = -\infty$, where $\epsilon_k = \epsilon_-$ and $w_{j_+}^{i_-}(\sigma^k)$ are defined as in Algorithm 3.1. Thus, it follows that there exists a unique $\beta_k^* \in [0, 1/(nw_{j_+}^{i_-}(\sigma^k)))$ such that $\Delta_k'(\beta_k^*) = 0$. This completes the first part of the proof.

From (20a), we have

$$\Delta_k'(\beta) \geq \frac{n}{2(1+\beta)} - \frac{1}{2}\sum_{j=1}^{n} \frac{nw_j^{i_-}(\sigma^k)}{1-\beta nw_{j_+}^{i_-}(\sigma^k)} = \frac{n}{2(1+\beta)} - \frac{n(1-\epsilon_k)}{2(1-\beta nw_{j_+}^{i_-}(\sigma^k))}.$$

It follows from this inequality that

$$\Delta_k'(\hat{\beta}_k) \geq \frac{1}{2}\left(\frac{n}{1+\hat{\beta}_k} - \frac{n(1-\epsilon_k)}{1-\hat{\beta}_k nw_{j_+}^{i_-}(\sigma^k)}\right) = 0 = \Delta_k'(\beta_k^*). \tag{21}$$

By solving Eq. (21), we obtain

$$\beta_k^* \geq \hat{\beta}_k = \frac{\epsilon_k}{1-\epsilon_k + nw_{j_+}^{i_-}(\sigma^k)}. \tag{22}$$

In addition, considering the feasibility of σ^{k+1} in "an away step", we complete the proof of (19). □

Remark 3.1. Compared with the proof of Lemma 2.1, it is easy to verify that for "an away step", $\Delta_k'(\beta)$ is no strictly convex function on $[0, \beta_k^*]$ in Lemma 3.1. Therefore, we cannot use the method of tangent equation for solving a lower bound $\hat{\beta}_k$ of the maximizer β_k^*.

The following analysis is similar to that of [7] for a similar algorithm that computes an approximation to the MVEE problem. According to the different values of $\hat{\beta}_k$, we can distinguish the three kinds of iterations in Algorithm 3.1. If $\hat{\beta}_k = \epsilon_k/[1 + n\sum_{j=1}^{n}(w_j^{i_+}(\sigma^k))^2]$, we call the kth iteration an increase-iteration, because only one component $\sigma_{i_+}^k$ of σ^k is increased. If $\hat{\beta}_k = \epsilon_k/[1-\epsilon_k + nw_{j_+}^{i_-}(\sigma^k)]$, we call it a decrease-iteration, since only one positive component $\sigma_{i_-}^k$ of σ^k is decreased. In addition, the core set remains unchanged at a decrease-iteration. Finally, if $\hat{\beta}_k = \sigma_{i_-}^k/(1-\sigma_{i_-}^k)$, we call it a drop-iteration, because the positive component $\sigma_{i_-}^k$ of σ^k becomes zero and the corresponding point x^{i_-} is dropped from the core set.

The following lemma gives a lower bound on the improvement of $\Delta_k(\hat{\beta}_k)$ at each increase- or decrease-iteration.

Lemma 3.2. *Let β_k^* be the unique maximizer of $\Delta_k(\beta)$ and $\hat{\beta}_k$ be given by (12) or (22). Then, at each increase- or decrease-iteration, we have*

$$\Delta_k(\beta_k^*) \geq \Delta_k(\hat{\beta}_k) \geq \begin{cases} 1/18, & \text{if } \epsilon_k \geq 1, \\ \epsilon_k^2/18, & \text{if } \epsilon_k < 1. \end{cases} \tag{23}$$

Proof. At an increase-iteration, the result directly follows from Lemma 2.2. At a decrease-iteration, we obtain from (20a) and (21) that

$$\frac{1}{1+\hat{\beta}_k} \geq \sum_{j=1}^{n} \frac{w_j^{i_-}(\sigma^k)}{1-\hat{\beta}_k nw_j^{i_-}(\sigma^k)}. \tag{24}$$

By (18) and (22), we have

$$\begin{aligned}
\Delta_k(\beta_k^*) &\geq \Delta_k(\hat{\beta}_k) \\
&= \frac{n}{2}\log(1+\hat{\beta}_k) + \frac{1}{2}\sum_{j=1}^{n}\log(1-\hat{\beta}_k nw_j^{i_-}(\sigma^k)) \\
&\geq \frac{n}{2}\frac{\hat{\beta}_k}{1+\hat{\beta}_k} + \frac{1}{2}\sum_{j=1}^{n}\log(1-\hat{\beta}_k nw_j^{i_-}(\sigma^k)) \\
&\geq \frac{1}{2}\sum_{j=1}^{n}\left(\log(1-\hat{\beta}_k nw_j^{i_-}(\sigma^k)) + \frac{\hat{\beta}_k nw_j^{i_-}(\sigma^k)}{1-\hat{\beta}_k nw_j^{i_-}(\sigma^k)}\right),
\end{aligned}$$

where we used $\log(1+x) \geq x/(1+x)$ for $x > 0$ and (24) to derive the second and the third inequalities, respectively.

Table 1
Arithmetic average of the CPU time, core set size and the number of iterations for N2.1, Algorithm 2.1, N3.1 and Algorithm 3.1 with $\epsilon = 10^{-3}$.

Dimensions		Time/s				Core set size				# Iterations			
n	m	N2.1	2.1	N3.1	3.1	N2.1	2.1	N3.1	3.1	N2.1	2.1	N3.1	3.1
10	5,000	3.80	3.22	0.22	0.10	31.9	32.1	14.4	14.6	11 935	12 262	250	324
10	10,000	5.63	5.24	0.23	0.15	31.2	31.2	13.5	13.6	11 884	11 831	212	251
20	10,000	20.38	18.64	0.70	0.46	61.1	60.6	23.6	23.6	26 552	26 559	440	551
20	20,000	76.65	74.33	1.95	2.35	64.5	63.4	25.5	25.5	27 142	27 007	534	784
40	20,000	280.71	277.73	3.24	4.19	114.9	115.0	37.6	37.6	56 277	56 209	551	802
40	40,000	576.36	573.19	**6.60**	9.44	120.1	119.5	40.2	40.2	57 180	57 262	587	897

It is easy to verify that $f_3(x) := \log(1-x)+x/(1-x)$ is strictly increasing for $x \in [0, 1)$ and $f_3(x) \geq (1/2)x^2$ for $x \in [0, 1)$. Since we always have $\epsilon_k < 1$ for a decrease-iteration, we obtain $\hat{\beta}_k n w_{j_+}^{i_-}(\sigma^k) = \epsilon_k n w_{j_+}^{i_-}(\sigma^k)/(1-\epsilon_k+n w_{j_+}^{i_-}(\sigma^k)) < 1$, where $\hat{\beta}_k$ and $w_{j_+}^{i_-}(\sigma^k)$ are defined as in Algorithm 3.1. Therefore, we have

$$\Delta_k(\hat{\beta}_k) \geq \frac{1}{2}\sum_{j=1}^{n}\frac{1}{2}(\hat{\beta}_k n w_j^{i_-}(\sigma^k))^2 = \frac{n^2}{4}\frac{\epsilon_k^2\sum_{j=1}^{n}(w_j^{i_-}(\sigma^k))^2}{(1-\epsilon_k+n w_{j_+}^{i_-}(\sigma^k))^2} = \frac{\epsilon_k^2}{4}\frac{\sum_{j=1}^{n}(w_j^{i_-}(\sigma^k))^2}{((1-\epsilon_k)/n+w_{j_+}^{i_-}(\sigma^k))^2}.$$

Using the definitions of $\epsilon_k := \epsilon_-$ and $w_{j_+}^{i_-}(\sigma^k)$ in Algorithm 3.1, we obtain

$$1-\epsilon_k = \sum_{j=1}^{n} w_j^{i_-}(\sigma^k) \leq n w_{j_+}^{i_-}(\sigma^k) \quad \text{and} \quad (w_{j_+}^{i_-}(\sigma^k))^2 \leq \sum_{j=1}^{n}(w_j^{i_-}(\sigma^k))^2.$$

Thus, we have $\Delta_k(\hat{\beta}_k) \geq \frac{\epsilon_k^2}{4}\frac{(w_{j_+}^{i_-}(\sigma^k))^2}{(2w_{j_+}^{i_-}(\sigma^k))^2} = \frac{\epsilon_k^2}{16} \geq \frac{\epsilon_k^2}{18}$. This completes the proof of (23). □

Remark 3.2. Lemma 3.2 establishes that Algorithm 3.1 makes at least as much improvement of $\Delta_k(\hat{\beta}_k)$ as Algorithm 2.1 at each increase- or decrease-iteration. At each drop-iteration, it is difficult to find a positive lower bound on $\Delta_k(\hat{\beta}_k) \geq 0$. Using the similar technique as in [7], each drop-iteration can be paired with a previous increase-iteration where $\sigma_{i_-}^k$ was increased from zero, except for those where $\sigma_{i_-}^k$ was positive at the initial iteration and was decreased to zero for the first time. Therefore, we can double the iteration count in the analysis of Algorithm 2.1 to establish that Algorithm 3.1 computes a $(1+\epsilon)$-approximation to MVAE(S) in at most twice as many iterations as that required by Algorithm 2.1. In addition, each iteration of Algorithm 3.1 still requires $O(mn)$ arithmetic operations, which implies that the overall complexity result of Algorithm 3.1 also remains the same as that of Algorithm 2.1. Finally, the core set result is also unaffected. However, since Algorithm 3.1 uses the idea of "dropping points", it has the potential to compute smaller core sets than those returned by Algorithm 2.1 in practice. We summarize these results in the following theorem.

Theorem 3.1. *Let $n \geq 2, \epsilon \in (0, 1)$. Algorithm 3.1 computes a $(1+\epsilon)$-approximation to* MVAE(S) *in $O(mn^3/\epsilon)$ arithmetic operations and returns an ϵ-core set $X \subseteq S$ such that $|X| = O(n^2/\epsilon)$.*

4. Numerical experiments

In this section, we report some numerical results for computing a $(1+\epsilon)$-approximation to MVAE(S). We implemented Algorithms 2.1 and 3.1 in MATLAB. For the purposes of comparison, we also implemented the numerical methods of Algorithms 2.1 and 3.1, which are denoted by N2.1 and N3.1 in Tables 1 and 2, respectively. Since the function $\Delta_k'(\beta)$ is strictly convex on $[0, \beta_k^*]$, we use Newton's method to compute the approximate β_k^* numerically for Algorithm 2.1 and increase-iterations of Algorithm 3.1. For decrease-iterations of Algorithm 3.1, we use "fzero function" in MATLAB, which finds a zero of a function in a given interval.

In our experiments, all computations were conducted on a Pentium IV processor with a speed of 3.0 GHz and 1GB RAM, running under Windows XP and MATLAB version 7.5.0.342 (R2007b). The data sets were randomly generated using the standard normal distribution with sizes (n, m) (see, Tables 1 and 2). For each fixed (n, m), ten different problem instances were randomly generated. The numerical results are reported in terms of the averages over these instances.

In our numerical experiments, we tested Algorithm 2.1 and N2.1 for the large-scale problem instances with a small ϵ (e.g. $\epsilon = 10^{-4}$), whose running speed is very slow. Therefore, we implemented these algorithms for a low accuracy $\epsilon = 10^{-3}$ and reported the results in Table 1, which include comparisons of the CPU time, core set size and the number of iterations, respectively.

As illustrated by Table 1, the CPU time and the number of iterations of N3.1 and Algorithm 3.1 are much less than those of N2.1 and Algorithm 2.1. In particular, for the large-scale data sets such as $n = 40$, $m = 40,000$, the CPU time of N3.1 is only

Table 2
Arithmetic average of the CPU time, core set size and the number of iterations for N3.1 and Algorithm 3.1 with $\epsilon = 10^{-7}$.

Dimensions		Time/s		Core set size		#Iterations	
n	m	N3.1	3.1	N3.1	3.1	N3.1	3.1
10	5,000	0.62	0.35	14.1	14.1	835	1071
10	10,000	1.02	0.74	15.0	15.0	1006	1326
20	10,000	1.74	1.31	24.2	24.2	1153	1606
20	20,000	4.14	4.99	23.4	23.4	1157	1681
40	20,000	9.21	13.09	38.8	38.8	1594	2511
40	40,000	16.79	25.76	39.4	39.4	1512	2466
100	100,000	**142.43**	262.77	78.1	78.1	2527	4777

a few seconds. In addition, it is surprising to note that the CPU time and the number of iterations of Algorithm 2.1 is very close to those of N2.1, which numerically computes an almost exact maximizer β_k^* using Newton's method. This indicates that $\hat{\beta}_k$ of Algorithm 2.1 is very close to the exact maximizer β_k^*. Note that the approximate β_k^* computed numerically using Newton's method need a little additional time, so the CPU time of Algorithm 2.1 is almost always less than that of the numerical method. However, as the data size increases, the CPU time and the number of iterations of Algorithm 3.1 become larger than those of N3.1, which implies that $\hat{\beta}_k$ of decrease-iterations in Algorithm 3.1 has a bigger gap than β_k^*.

In terms of core set, core set size of N2.1 and Algorithm 2.1 (N3.1 and Algorithm 3.1) remains almost the same. However, core set size of Algorithm 3.1 is only $1/3$ to $1/2$ of that of Algorithm 2.1, which indicates that Algorithm 3.1 is indeed able to compute smaller core sets in practice.

In Table 2, we present the performances of N3.1 and Algorithm 3.1 for a high accuracy $\epsilon = 10^{-7}$. As indicated by Table 2, the core set size of Algorithm 3.1 remains the same as N3.1. The CPU time and the number of iterations of the two methods have the similar relations as Table 1. In addition, the CPU time of N3.1 is very few. For example, it solves very large instances ($n = 100$, $m = 100{,}000$) in about 2 min.

In a word, the numerical results indicate that our modifications have a very good performance, which is also consistent with results of our theoretical analysis.

5. Final remarks

In this paper, we proposed two modifications of the KY algorithm [5] that computes a $(1 + \epsilon)$-approximation to the minimum volume axis-aligned ellipsoid enclosing a given finite set of points. In the KY algorithm, the complexity bound and the core set size are $O(mn^5/\epsilon)$ and $O(n^4/\epsilon)$ for $\epsilon \in (0, 1)$, respectively. The modified algorithms reduce them to $O(mn^3/\epsilon)$ and $O(n^2/\epsilon)$ for $\epsilon \in (0, 1)$, respectively. Our numerical results show that our second modification exhibits significantly better performance, and it indeed computes smaller core sets in practice. In particular, our second modified algorithm is very efficient for solving large-scale problem with a high accuracy.

There are several interesting theoretical and practical problems that are motivated by our modifications. In Section 2, since the function $\Delta_k'(\beta)$ is strictly convex on $[0, \beta_k^*]$ by Lemma 2.1, the proposed line search method (the tangent equation method) may be considered as a single Newton step maximizing (11). Is there an "optimal", or at least "better" number of Newton steps in the numerical method that leads to a faster implementation than the proposed algorithm? It is worth exploring whether such a better implementation exists computationally. Theoretically, is it possible to show that such an approach leads to better complexity results by conducting multiple Newton steps? In addition, whether the proposed algorithms can be combined with an active set strategy [6] that speeds up the computations? Also, it is also worth exploring whether linear convergence of Algorithm 3.1 exists according to [1]. Other interesting problems have not yet been resolved the problem proposed in [5]. These are topics of our further study in the near future.

Acknowledgements

The authors cordially thank two anonymous referees for their valuable comments which led to the improvement of this paper.

References

[1] S.D. Ahipasaoglu, P. Sun, M.J. Todd, Linear convergence of a modified Frank–Wolfe algorithm for computing minimum volume enclosing ellipsoids, Optimization Methods and Software 23 (2008) 5–19.
[2] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval Research Logistics Quarterly 3 (1956) 95–110.
[3] L.G. Khachiyan, Rounding of polytopes in the real number model of computation, Mathematics of Operations Research 21 (1996) 307–320.
[4] P. Kumar, E.A. Yıldırım, Minimum volume enclosing ellipsoids and core sets, Journal of Optimization Theory and Applications 126 (1) (2005) 1–21.
[5] P. Kumar, E.A. Yıldırım, Computing minimum volume enclosing axis-aligned ellipsoids, Journal of Optimization Theory and Applications 136 (2) (2008) 211–228.
[6] P. Sun, R.M. Freund, Computation of minimum volume covering ellipsoids, Operations Research 52 (2004) 690–706.
[7] M.J. Todd, E.A. Yıldırım, On Khachiyan's algorithm for the computation of minimum volume enclosing ellipsoids, Discrete Applied Mathematics 155 (2007) 1731–1744.